**LAB REQUIREMENTS: ruby 2.3.1p112 ; perl 5 ; TCL (8.6.0+9).**

1.**Write a ruby script to create a new string which is n copies of a given string where n is a non-negative integer**

**PROGRAM:**

puts "Enter a string:"

a=gets.chomp

puts "Enter no of copies:"

b=gets.chomp.to\_i

puts "New string is: #{(a\*b)} "

**OUTPUT:**

Enter a string:

sonia

Enter no of copies:

2

New string is: soniasonia

2**.Write a ruby ruby script which accepts the radius of acircle from the user and compute the parameter and area**

**PROGRAM:**

puts "Enter radius of circle:"

radius=gets.chomp.to\_i

x=2\*3.14\*radius

y=3.14\*radius\*radius

puts "Paramater is: #{x}"

puts "Area is:#{y}"

**OUTPUT:**

Enter radius of circle:

4

Paramater is: 25.12

Area is:50.24

3.**write aruby script which accept the user’s first and last name and print them in reverse order with a space between them**

**PROGRAM:**

puts "Enter first name:"

fn=gets.chomp

puts "Enter last name:"

ls=gets.chomp

puts "reverse order: "

puts "#{ls} #{fn}"

**OUTPUT:**

Enter first name:

sonia

Enter last name:

kovuru

reverse order:

kovuru sonia

4.**write a ruby script to accept a file name from the user print extension of that**

**PROGRAM:**

puts "Enter file name:"

fname=gets.chomp

arr=fname.split('.')

puts "The extension is: #{arr[1]}"

**OUTPUT:**

Enter file name:

text.java

The extension is: java

5.**Write a ruby script to find greatest of three numbers**

**PROGRAM:**

puts "Enter num1:"

a=gets.chomp

puts "Enter num2:"

b=gets.chomp

puts"Enter num3:"

c=gets.chomp

if a>b && a>c

puts "num1 is greatest: #{a}"

elsif b>a && b>c

puts "num2 is greatest: #{b}"

else

puts "num3 is greatest: #{c}"

end

**OUTPUT:**

Enter num1:

5

Enter num2:

4

Enter num3:

3

num1 is greatest: 5

**6.Write a ruby script to print odd numbers from 10 to 1**

**PROGRAM:**

puts "odd numbers from 10 to 1:"

9.step 1,-2 do |x|

puts "#{x}"

end

**OUTPUT:**

odd numbers from 10 to 1:

9

7

5

3

1

**7.write a ruby script to check two integers and return if one of them is 20 otherwise return their sum**

**PROGRAM:**

puts "Enter a num1:"

a=gets.chomp.to\_i

puts "Enter a num2:"

b=gets.chomp.to\_i

if (a==20 or b==20)

puts("True")

else

add=a+b

puts "addition of two numbers: #{add}"

end

**OUTPUT:**

Enter a num1:

20

Enter a num2:

3

True

8.**write a ruby script to check two integers and return true if one of them is less than 0 and other is greater than 100**

**PROGRAM:**

puts "Enter a temp1:"

a=gets.chomp.to\_i

puts "Enter a temp2:"

b=gets.chomp.to\_i

if (a<0 ) && (b>100) || (b<0) && (a>100)

puts ("True")

end

**OUTPUT:**

Enter a temp1:

200

Enter a temp2:

-4

True

9.**Write a ruby script to print elements of given array**

**PROGRAM:**

array=["s","o","n","i","a"]

for i in array

puts i

end

**OUTPUT:**

s

o

n

i

a

10.**write a ruby script to retrieve the total marks where subject name and marks of a student stored in a hash**

**PROGRAM:**

student\_marks={DAA:60,ML:50,SL:60}

total\_marks=0

student\_marks.each {|key,value|

total\_marks+=value

}

print "Total marks:"+total\_marks.to\_s

**OUTPUT:**

Total marks:170

11.**Write a TCL script to find the factorial of a number**

**PROGRAM:**

puts "Enter a number"

gets stdin b

proc factorial {number} {

if {$number <= 1} {

return 1

} else {

return [expr $number \* [factorial [expr $number - 1]]]

}

}

puts [factorial $b]

**OUTPUT:**

Enter a number

5

120

12.**Write a TCL script that multiplies the numbers from 1 to 10**

**PROGRAM:**

puts "Enter a number"

gets stdin b

puts "multiplies of given number from 1 t0 10:"

for { set i 1} { $i<=10} {incr i} {

set mul [expr $b\*$i]

puts $b\*$i=$mul

}

**OUTPUT:**

Enter a number

4

multiplies of given number from 1 t0 10:

4\*1=4

4\*2=8

4\*3=12

4\*4=16

4\*5=20

4\*6=24

4\*7=28

4\*8=32

4\*9=36

4\*10=40

13.**Write a TCL script for sorting a list using a comparison function**

**PROGRAM:**

set color [list red green blue]

set color [lsort $color]

puts $color

**OUTPUT:**

blue green red

14.**Write a TCL script to (i) create a list (ii)append elements to the list (iii)Traverse the list (iv)Concatenate the list**

**PROGRAM:**

puts "creating a list:"

set flower [list rosy lily]

puts $flower

puts "appending a list:"

lappend flower tulip

puts $flower

puts "traversing a list:"

foreach i $flower {

puts $i

}

puts "concatenating a list:"

set flow2 [list lotus]

set res [concat $flower $flow2]

puts $res

**OUTPUT:**

creating a list:

rosy lily

appending a list:

rosy lily tulip

traversing a list:

rosy

lily

tulip

concatenating a list:

rosy lily tulip lotus

15.**write a tcl script to comparing the file modified time**

**PROGRAM:**

set tclfiles [glob \*.tcl]

puts "Name - data of last modification:"

foreach f $tclfiles {

puts "$f - [clock format [file mtime $f] -format %x ]"

puts "$f - [clock format [file atime $f] -format %x ]"

}

**OUTPUT:**

Name - data of last modification:

tcl15.tcl - 08/03/2021

tcl15.tcl - 08/03/2021

tc15.tcl - 08/03/2021

tc15.tcl - 08/03/2021

tcl4.tcl - 08/03/2021

tcl4.tcl - 08/04/2021

hello.tcl - 08/06/2021

hello.tcl - 08/06/2021

tcl16.tcl - 08/06/2021

tcl16.tcl - 08/06/2021

native.tcl - 08/06/2021

native.tcl - 08/06/2021

tcl3.tcl - 08/03/2021

tcl3.tcl - 08/06/2021

tclf.tcl - 08/04/2021

tclf.tcl - 08/04/2021

tcl5.tcl - 08/03/2021

tcl5.tcl - 08/06/2021

tcl1.tcl - 07/31/2021

tcl1.tcl - 08/04/2021

tcl2.tcl - 07/31/2021

tcl2.tcl - 08/04/2021

tcl33.tcl - 08/03/2021

tcl33.tcl - 08/04/2021

tcl6.tcl - 08/06/2021

tcl6.tcl – 08/06/2021

16.**write a tcl script to copy a file and translate into native format**

**PROGRAM:**

proc f\_Copy {src dest} {

if [file isdirectory $src] {

file mkdir $dest

foreach f [global [file join $src \*]] {

f\_copy $f [file join $dest [file tail $f]]

}

return

}

if [file isdirectory $dest] {

set dest [file join $dest [file tail $src]]

}

set in [open $src r]

set out [open $dest w+]

puts $out [read $in]

close $out;

close $in;

}

**OUTPUT:**

set color [list red green blue]

set color [lsort $color]

puts $color

puts [f\_Copy tcl3.tcl native.tcl]

OUTPUT WILL BE SAVED IN NATIVE.TCL

17.

A)**write a perl script to find the largest number among three numbers**

**PROGRAM:**

#!/usr/bin/perl

print "Enter num1:";

$a=<stdin>;

print "Enter num2:";

$b=<stdin>;

print "Enter num3:";

$c=<stdin>;

if ($a>$b and $a>$c)

{

print "$a is greatest";

}

elsif ($b>$a and $b>$c)

{

print "$b is greatest";

}

else

{

print "$c is greatest";

}

**OUTPUT:**

Enter num1:5

Enter num2:4

Enter num3:3

5

is greatest

B)**Write a perl script to print the multiplication tables from 1-10 using subroutines**

**PROGRAM:**

use strict;

use warnings;

sub table{

our @i=qw(1 2 3 4 5 6 7 8 9 10);

foreach my $j(@i){

multiplication($j);

}

}

sub multiplication{

my $num=$\_[0];

my @i=qw(1 2 3 4 5 6 7 8 9 10);

foreach my $j(@i){

my $res=$num\*$j;

print "$num \* $j=$res\n";

}

}

table();

**OUTPUT:**

1 \* 1=1

1 \* 2=2

1 \* 3=3

1 \* 4=4

1 \* 5=5

1 \* 6=6

1 \* 7=7

1 \* 8=8

1 \* 9=9

1 \* 10=10

2 \* 1=2

2 \* 2=4

2 \* 3=6

2 \* 4=8

2 \* 5=10

2 \* 6=12

2 \* 7=14

2 \* 8=16

2 \* 9=18

2 \* 10=20

3 \* 1=3

3 \* 2=6

3 \* 3=9

3 \* 4=12

3 \* 5=15

3 \* 6=18

3 \* 7=21

3 \* 8=24

3 \* 9=27

3 \* 10=30

4 \* 1=4

4 \* 2=8

4 \* 3=12

4 \* 4=16

4 \* 5=20

4 \* 6=24

4 \* 7=28

4 \* 8=32

4 \* 9=36

4 \* 10=40

5 \* 1=5

5 \* 2=10

5 \* 3=15

5 \* 4=20

5 \* 5=25

5 \* 6=30

5 \* 7=35

5 \* 8=40

5 \* 9=45

5 \* 10=50

6 \* 1=6

6 \* 2=12

6 \* 3=18

6 \* 4=24

6 \* 5=30

6 \* 6=36

6 \* 7=42

6 \* 8=48

6 \* 9=54

6 \* 10=60

7 \* 1=7

7 \* 2=14

7 \* 3=21

7 \* 4=28

7 \* 5=35

7 \* 6=42

7 \* 7=49

7 \* 8=56

7 \* 9=63

7 \* 10=70

8 \* 1=8

8 \* 2=16

8 \* 3=24

8 \* 4=32

8 \* 5=40

8 \* 6=48

8 \* 7=56

8 \* 8=64

8 \* 9=72

8 \* 10=80

9 \* 1=9

9 \* 2=18

9 \* 3=27

9 \* 4=36

9 \* 5=45

9 \* 6=54

9 \* 7=63

9 \* 8=72

9 \* 9=81

9 \* 10=90

10 \* 1=10

10 \* 2=20

10 \* 3=30

10 \* 4=40

10 \* 5=50

10 \* 6=60

10 \* 7=70

10 \* 8=80

10 \* 9=90

10 \* 10=100

18.**Write a perl program to implement the following list of manipulating functions**

**(a)shift**

**(b)unshift**

**(c)push**

**PROGRAM:**

@x = ('red','green','blue');

print "Original array: @x \n";

push(@x,'orange','yellow');

print "updated array : @x \n";

print "values returned by shift :",shift(@x);

print "\narray after shifting :@x\n";

print "No of elements returned by unshift:",unshift(@x, 'pink', 'purple');

print "\narray after unshift:@x\n";

**OUTPUT:**

Original array: red green blue

updated array : red green blue orange yellow

values returned by shift :red

array after shifting :green blue orange yellow

No of elements returned by unshift:6

array after unshift:pink purple green blue orange yellow

19.(a**)write a perl script to substitue a word with another word in a string**

**PROGRAM:**

$string ="hello world";

$string =~ s/world/INDIA/;

print "$string";

**OUTPUT:**

hello INDIA

B)**Write a perl script to validate IP address and email address**

**IP ADDRESS:**

#!/usr/bin/perl

print("Enter the IP Address you would like to validate - ");

my $ip = <STDIN>;

if($ip =~ /(\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})/)

{

$ip = $1;

}

chomp($ip);

if($ip =~ m/^(\d\d?\d?)\.(\d\d?\d?)\.(\d\d?\d?)\.(\d\d?\d?)$/)

{

print("\nIP address found - $ip\n");

if($1 <= 255 && $2 <= 255 && $3 <= 255 && $4 <= 255)

{

print("Each octet of an IP address is ",

"within the range - $1.$2.$3.$4\n");

print("\n-> $ip IP address accepted!\n");

}

else

{

print("Octet(s) out of range. ",

"Valid number range between 0-255\n");

}

}

else

{

print("IP Address $ip is not in a valid format\n");

}

**OUTPUT:**

Enter the IP Address you would like to validate - 192.168.100.37

IP address found - 192.168.100.37

Each octet of an IP address is within the range - 192.168.100.37

-> 192.168.100.37 IP address accepted!OUTPUT:

**EMAILADDRESS:**

**PROGRAM:**

use strict;

use warnings;

use 5.010;

use Email::Valid;

print "Enter email address:";

my $add = <> ;

my $address = Email::Valid->address($add);

say ($address ? "yes '$address' " : "no '$add' ");

**OUTPUT:**

Enter email address:sonia@gmail.com

yes ['sonia@gmail.com](mailto:'sonia@gmail.com)'

20.**write a perl script to print the file in reverse order using command line arguments**

**PROGRAM:**

perl -ne 'chomp;

> print scalar reverse;'

welcome

emoclew